CLAIMS

1. (Currently amended) A method for allocating core processor bandwidth, comprising:

detecting an interrupt service request;

inserting interrupt servicing instructions into an instruction queue mechanism in response to detecting the interrupt service request, wherein the instruction queue mechanism includes an instruction cache and an instruction fetch unit;

~~concurrently in-line staging~~ inserting the interrupt servicing instructions ~~inserted~~ into mainline program instructions within the instruction queue mechanism resulting in allocating core processor bandwidth between the interrupt servicing and mainline program instructions; and

~~executing the concurrently in-line staged mainline programs and the interrupt servicing instructions in the instruction queue mechanism.~~

decoding the mainline program and the interrupt servicing instructions into micro-ops;

executing the micro-ops in one or more out-of-order execution units;

recycling the executed micro-ops for optional re-execution in the one or more out-of-order execution units by retiring the executed micro-ops including those micro-ops representing the interrupt servicing instructions to the instruction cache.

2. (Canceled)

3. (Canceled)

4. (Currently amended) The method of claim ~~3~~ 1 comprising retiring the executed micro-ops to the instruction cache in order.

5. (Previously presented) The method of claim 1 where detecting comprises detecting plural interrupts by prioritizing the plural interrupts and inserting one or more instances of the interrupt servicing instructions into the instruction queue mechanism in accordance with one or more predefined interrupt servicing allocation criteria.

6. (Previously presented) The method of claim 5 where the one or more allocation criteria include the priority of the interrupts and the capacity of the processor to allocate bandwidth to interrupt servicing.

7. (Previously presented) The method of claim 6 where the prioritizing is dynamically responsive to changing allocation criteria.

8. (Previously presented) The method of claim 1 where the processor bandwidth is allocated to interrupt servicing without flushing the instruction queue mechanism.

9. (Previously presented) The method of claim 1 where the detecting is performed by an interrupt processor, which after the detecting, the method comprising:

at the interrupt processor, determining whether the detected interrupt service request is of a priority meeting one or more defined high-priority criteria and if so then signaling the processor to perform the inserting; and

at the core processor, responding to said signaling by performing said inserting and said processing.

10. (Currently amended) A method comprising:

detecting an interrupt service request;

inserting interrupt servicing instructions into an instruction queue mechanism in response to detecting the interrupt service request where inserting the interrupt servicing instructions into the instruction queue mechanism results in a core processor ~~concurrently in-line staging~~ allocating bandwidth between the interrupt servicing instructions and other program instructions;

executing the interrupt servicing instructions and the other program instructions within the instruction queue mechanism;

determining that a natural processor context switch is imminent, said determining including analyzing one or more instructions within the instruction queue mechanism for context switch-inducing instructions;

signaling the interrupt processor to make ready the highest priority interrupt service request; and

signaling the instruction queue mechanism to fetch the readied interrupt service request in advance of a context switch responsive to the determining;

where the detecting is performed by an interrupt processor.

11. (Currently amended) A processor, comprising:

an instruction cache;

~~a fetch unit to fetch mainline program instructions from the instruction cache;~~

an interrupt handler to detect interrupt servicing instructions;

a fetch unit to fetch mainline program and interrupt servicing instructions from the instruction cache;

a decode unit to decode mainline program and interrupt servicing instructions into micro-op~~eode~~s;

a dispatch unit to schedule the micro-op~~eode~~s for execution;

an execute unit to execute the micro-op~~eode~~s; ~~and~~

a retirement unit to retire executed micro-ops including the micro-ops corresponding to the interrupt instructions back to the instruction cache; and

where the ~~decode unit inserts~~ interrupt handler signals the fetch and decode units to insert micro-op~~eode~~s corresponding to interrupt servicing instructions into an instruction sequence within the instruction cache without flushing the instruction cache.

12. (Currently amended) The processor of claim 11

where the processor supports plural hardware interrupt inputs;

where the interrupt-handler includes an interrupt concentrator to determine priority among the plural hardware interrupt inputs and to schedule the plural hardware interrupt inputs; and

where the interrupt-handler is adapted to instruct~~s~~ the fetch and decode units such that plural instances of such decoded micro-op~~eode~~s are inserted, each instance representing one or more corresponding sets of interrupt servicing instructions, into ~~the~~ a normal instruction sequence for serial scheduling and execution of the plural instances of such decoded micro-op~~eode~~s by the dispatch and execute units in accordance with ~~the~~ a determined priority of the plural hardware interrupt inputs.

13. (Currently amended) A processor comprising:

an instruction cache;

a fetch unit to fetch mainline program instructions and interrupt servicing instructions from the instruction cache;

an interrupt handler to detect interrupt servicing instructions;

a decode unit to decode the mainline program and interrupt servicing instructions into micro-op~~code~~s;

a dispatch unit to schedule the micro-op~~code~~s for execution;

an execute unit to execute the micro-op~~code~~s;

a context switch prediction unit coupled with the fetch and decode units to predict a naturally occurring context switch and to signal the interrupt handler upon such prediction, the context switch prediction unit being adapted to analyze one or more instructions within a context switch inducing instruction cache;

where the ~~decode unit inserts~~ interrupt handler signals the fetch and decode units to insert micro-op~~code~~s corresponding to interrupt servicing instructions into an instruction sequence within the instruction cache without flushing the instruction cache; and

where the interrupt handler is coupled to the prediction unit instructing the fetch and decode units.

14. (Currently amended) An apparatus, comprising:

an instruction queue mechanism to stage ~~p-code representative of~~ mainline and interrupt instructions for execution;

a plurality of interrupt inputs corresponding to a plurality of hardware input/output devices;

interrupt priority logic to prioritize the plurality of interrupt inputs;

bandwidth allocation logic to allocate processing resources to service the plurality of interrupts according to their relative priority;

a decoder to generate ~~the~~ p-code representative of the mainline and interrupt instructions responsive to the plurality of interrupts and the bandwidth allocation logic;

a p-code insertion mechanism to insert the p-code representative of the interrupt instructions into the p-code representative of the mainline instructions in ~~the~~ an instruction stream ~~queue mechanism~~ according to the processing resources allocated to the plurality of interrupt inputs;

a p-code processor coupled to the instruction queue mechanism and to the insertion mechanism to execute the p-code representative of the mainline and interrupt instructions in the instruction stream ~~queue mechanism~~;

a retirement unit to retire executed p-code representative of the mainline and interrupt instructions back to an ~~the~~ instruction ~~queue mechanism~~ cache.

15. (Previously presented) The apparatus of claim 14,

where the instruction queue mechanism includes an instruction cache and an in-order instruction unit to perform branch predictions; and

where the p-code processor includes a dispatch and out-of-order execution units to schedule and execute the p-code representative of the mainline and interrupt instructions in parallel among one or more execution ports.

16. (Canceled)

17. (Previously presented) The apparatus of claim 15 where the in-order instruction unit includes said allocation logic.

18. (Previously presented) The apparatus of claim 17 where the allocation logic is coupled to a computer operating system.

19. (Previously presented) The apparatus of claim 14 where the priority logic is coupled to a computer operating system.

20. (Previously presented) The apparatus of claim 14 where the allocation logic is coupled to a current-usage model.

21. (Previously presented) The apparatus of claim 14 where the priority logic is coupled to a current-usage model.

22. (Currently amended) A machine-readable medium comprising a computer-readable medium containing instructions, that, when executed by a machine cause the machine to perform a method comprising:

detecting an interrupt service request;

inserting interrupt servicing instructions responsive to the interrupt service request into mainline program instructions within an instruction queue mechanism;

~~decoding interrupt servicing instructions associated with the detected interrupt service request into interrupt servicing p-code;~~

~~inserting the interrupt servicing p-code responsive to the interrupt service request into mainline program p-code in an instruction queue mechanism;~~

~~processing the interrupt servicing p-code inserted in the mainline program p-code within the instruction queue mechanism; and~~

processing instructions within the instruction queue mechanism including the inserted interrupt servicing instructions; and

signaling after detecting an impending natural context switch represented by a recognized instruction sequence ~~in the interrupt servicing p-code inserted in the mainline program p-code~~ within the instruction queue mechanism;

where detecting an impending natural context switch is predicated on an instruction stream within the instruction queue mechanism of the machine.

23. (Previously presented) The article of claim 22 comprising:

signaling with a vector representing an interrupt service request upon determining that an interrupt service request of a priority that exceeds a given threshold has occurred; and

fetching interrupt service instructions from memory in accordance with the vector representing the determined priority interrupt service request.

24. (Canceled)

25. (Currently amended) A computer system, comprising:

one or more input/output (I/O) devices to generate one or more hardware interrupts;

an interrupt concentrator to rank and queue the interrupts into an ordered interrupt array;

a core processor coupled to the interrupt concentrator, the core processor including an instruction cache, fetch unit, and a decode unit, the fetch unit to fetch instructions from the instruction cache and the decode unit to decode ~~the~~ interrupt service instructions and other program instructions into micro-op~~eode~~s, the core processor including a dispatch unit to schedule the micro-op~~eode~~s, and an execute unit having one or more execution ports to execute the micro-op~~eode~~s in the one or more execution ports; and

an interrupt-handler responsive to the one or more hardware interrupts, the interrupt-handler instructing the fetch and decode units to insert into an instruction sequence decoded micro-op~~eode~~s representing interrupt servicing instructions for scheduling and execution by the dispatch and execute units, wherein the instruction sequence also comprises decoded micro-ops representing other program instructions.

26. (Currently amended) The computer system of claim 25 which supports plural hardware interrupt inputs, where the interrupt-handler includes ~~an~~ the interrupt concentrator to determine priority among and to schedule the plural hardware interrupts, the interrupt-handler instructing the fetch and decode units to insert plural instances of such decoded micro-op~~eode~~s, each instance representing one or more corresponding sets of interrupt servicing instructions, into the normal instruction sequence for serial scheduling and execution of the plural instances of such decoded micro-op~~eode~~s by the dispatch and execute units in accordance with the determined priority of said plural hardware interrupts.

27. (Currently amended) The computer system of claim 25 comprising:

a context switch prediction unit coupled with the fetch and decode units to predict a naturally occurring context switch represented by a recognized instruction sequence within the instruction cache and to signal the ~~interrupt-handling mechanism~~ interrupt-handler upon such prediction;

where the ~~interrupt-handling mechanism~~ interrupt-handler is coupled to such signaling from the prediction unit instructing the fetch and decode units.

28. (Currently amended) A method, comprising:

detecting an interrupt service request;

inserting micro-ops representative of interrupt servicing instructions responsive to the interrupt service request into an instruction queue mechanism in such manner that the inserted micro-ops representative of the interrupt servicing instructions intervene micro-ops representative of mainline program instructions within ~~a~~ the instruction queue thereby allocating core processor bandwidth between the inserted micro-ops representative of the interrupt servicing instructions and the micro-ops representative of the mainline program instructions;

processing the micro-ops representative of the interrupt servicing inserted into the micro-ops representative of the mainline program instructions within the instruction queue mechanism, where core processor bandwidth allocation between the micro-ops representative of the mainline program instructions and the inserted micro-ops representative of the interrupt servicing instructions is achieved by concurrent in-line staging of the same within the instruction queue mechanism without first flushing the instruction queue mechanism of its contents; and

re-processing the micro-ops representative of the interrupt servicing instructions inserted into the micro-ops representative of the mainline program instructions in the one or more out-of-order execution units by reordering and retiring the executed micro-op~~code~~s including those micro-op~~code~~s representing the inserted interrupt servicing instructions to an instruction cache.